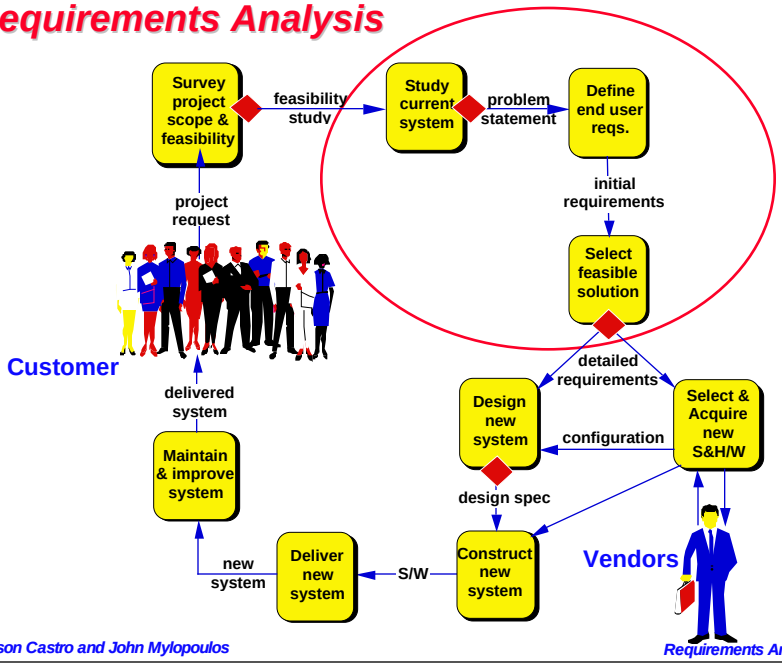
VIII. Requirements Analysis

Functional and Non-Functional Requirements
Customer-Driven vs Market-Driven vs User-Driven Projects
Visual Modeling
Use Cases
The State of Practice

Requirements Analysis



What Are Requirements?

"...Requirements definition is a careful assessment of the needs that a system is to fulfill...must say **why** a system is needed, based on current and foreseen conditions, which may be internal operations or an external market...must say **what** system features will serve and satisfy this context...must also say **how** the system is to be constructed..."
[Ross77]

- Requirements represent a specification for the new system.
- Effectively a "contract" between client and developer
- We usually distinguish between
  - ✓ Functional requirements, which describe functions that the new system must support;
  - ✓ Non-functional (or, quality) requirements, which impose global constraints on the system;

Functional Requirements

- Describe the processing required from the new system;
- Describe the inputs into the system from paper forms and documents, from interactions with people, such as email messages, and from other systems;
- Also describe the outputs that are expected from the system in the form of printed documents, screen displays and data transfers to other systems,
- Finally, they describe the data that must be held in, and managed by the system so that it can fulfill its required functions.

Functional requirements describe the system with respect to its environment, NOT its internal workings!

Non-Functional Requirements

- Describe aspects of the system that are concerned with how well it supports the functional requirements (hence the name non-functional, or quality requirements).
- This description may include:
  - ✓ Performance criteria such as desired response times for updating data in the system or retrieving data from the system;
  - ✓ Reliability requirements, e.g., the system must crash on average once every 6 months;
  - ✓ Security considerations, e.g., access rights for different groups of users;
  - ✓ Standards the working systems should meet;
  - ✓ Usability requirements, such as: users will be able to use the system after 2 days of training;
  - ✓ …more...

Importance of Requirements Analysis

- Most errors (54%) are detected after coding and testing.
- Almost half of all errors in software (45%) are in requirements and design.
- Most errors made during requirements analysis are non-clerical (77%) and may arise because of incorrect facts, misunderstandings, inconsistencies, omissions and ambiguities.
- Requirements errors can be detected, because inspection techniques have proven most effective for any software, and inspection techniques can be applied to requirements as well as design and code.

Relative Cost of Repairs

Stage	Relative Cost
Requirements	0.1 - 0.2
Preliminary Design	0.5
Coding	1.0
Unit Testing	2.0
Acceptance Test	5.0
Maintenance	20.0

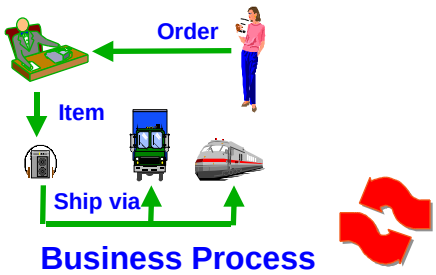
Customer- vs Market- vs User-Driven Projects

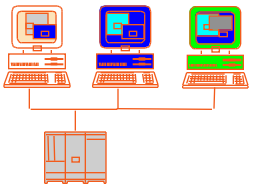
- Customer-driven projects involve a customer who needs a system that solves a particular problem; customer-driven projects often are one-of projects
- Market-driven projects involve a developer who decides to develop a (generic) system that is to be sold in the market; often hard to determine for such projects what the customer really wants (...or, for that matter, who the customer really is)
- (Coming soon) User-driven projects involve a system which is used, for a fee, by a number of users; owner of the system has to evolve it according to user demands
- The concept of software is evolving from that of a custom-built artifact, to that of a commodity that you buy, and soon to that of a resource that you use.

Visual Modeling

- Key problem: Have to give an unambiguous, easy to understand account of our understanding of an organization and how it works, also how the new system will fit in that organization.
- We can do so with English descriptions; but such descriptions are often cumbersome, incomplete, ambiguous and can lead to misunderstandings.
- As an alternative, we will use **visual models** of the proposed requirements which are easier to understand (if done properly) and can facilitate communication between the different stakeholders of an information system development project.
- We already saw some visual models when we were discussing organizations and software lifecycles.

What is Visual Modeling?





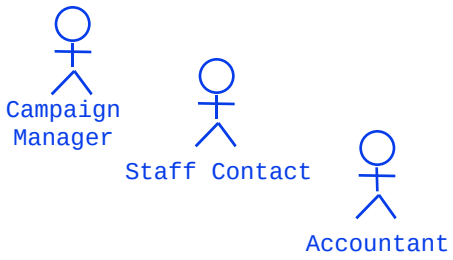
Visual Modeling is modeling using a standard graphical notation

Where Do We Start? Use Cases

- Use cases are descriptions of the functionality of the new system from a user's perspective.
- Used to show the functions to be provided by the system, also which users will use which functions.
- Developed by Ivar Jacobson and friends [Jacobson92].
- Part of the Unified Modeling Language (or, UML).

Actors

- Anything that needs to exchange information with the system
- Could be people, or other, external, systems.
- Define roles that users can play while using the system.

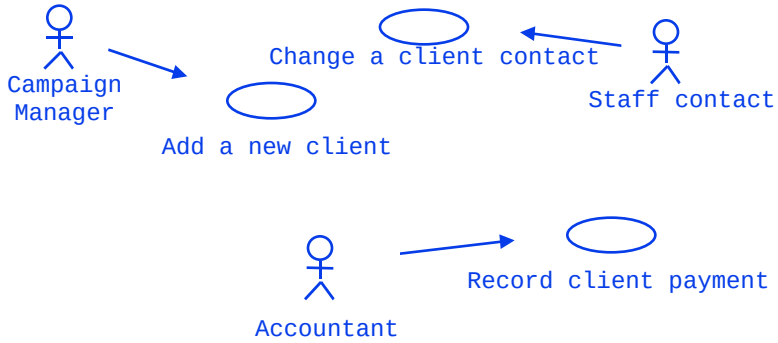


Use Cases

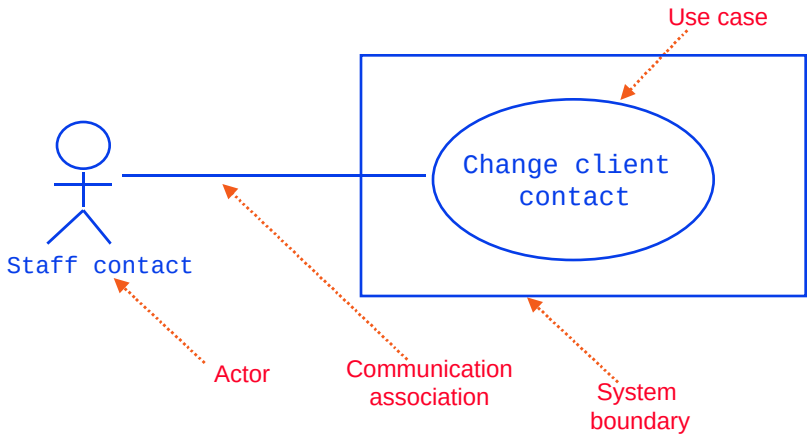
- A use case is a pattern of behavior which the new system is required to exhibit.
- Each use case is a sequence of related transactions performed by an actor and the system through a dialogue.
- To find use case, examine each actor and her needs, e.g.,
  - ✓Campaign Manager -- add a new client
  - ✓Staff Contact -- Change a client contact
  - ✓Accountant -- Record client payment

Use Case Diagrams

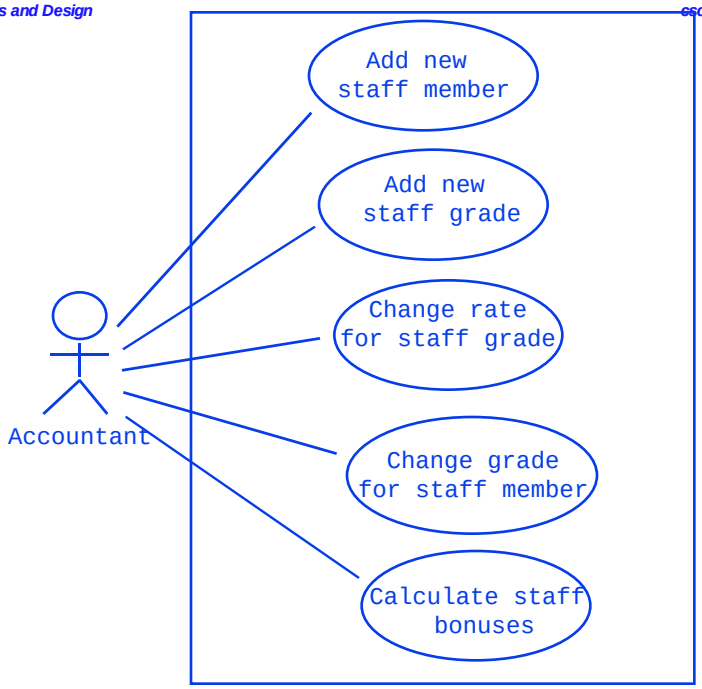
- Use case diagrams are created to capture the relationships between actors and use cases



Notation for Use Cases

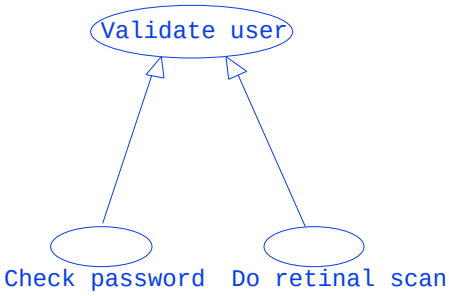


Agate Case Study



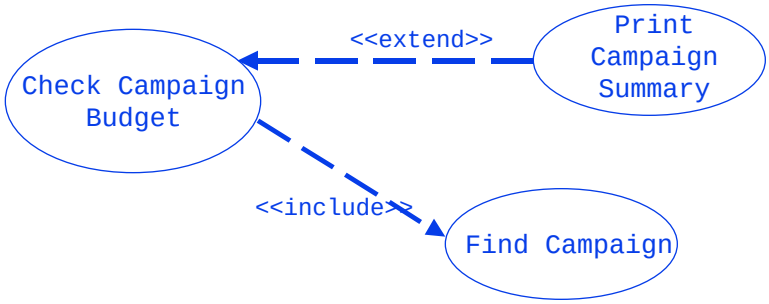
Use Case Relationships -- Generalization

- <<generalization>> is a relationship type for use cases in situations where one use case is a special case of another use case.
- For example,



<<extend>> and <<include>>

- <<extend>> implies that one use case adds behaviour to a base case; used to model a part of a use case that the user may see as optional system behavior; also models a separate sub-case which is executed conditionally.
- <<include>>: adds behavior to a base case (like a procedure call); used to avoid describing the same flow of events several times, by putting the common behavior in a use case of its own.



Finding Actors

- Actors can be identified by answering the following questions:
  - ✓Who will be a primary user of the system? (primary actor)
  - ✓Who will need support from the system to do her daily tasks?
  - ✓Who will maintain, administrate, keep the system working? (secondary actor)
  - ✓Which hardware devices does the system need?
  - ✓With which other systems does the system need to interact?
  - ✓Who or what has an interest in the results that the system produce ?
- Tip: don't consider only the users who directly use the system, but also others who need services from the system!

Finding Use Cases

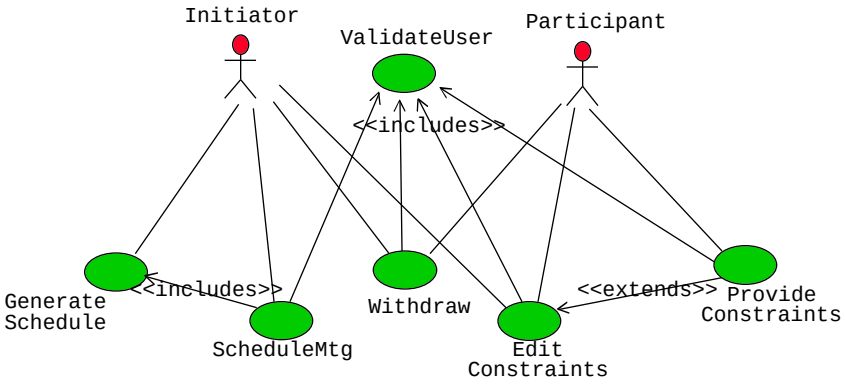
For each actor, ask the following questions:

- Which functions does the actor require from the system? What does the actor need to do ?
- Does the actor need to read, create, destroy, modify, or store some kinds of information in the system ?
- Does the actor have to be notified about events in the system? Or, does the actor need to notify the system about something? What do those events require in terms of system functionality?
- Could the actor's daily work be simplified or made more efficient through new functions provided by the system?

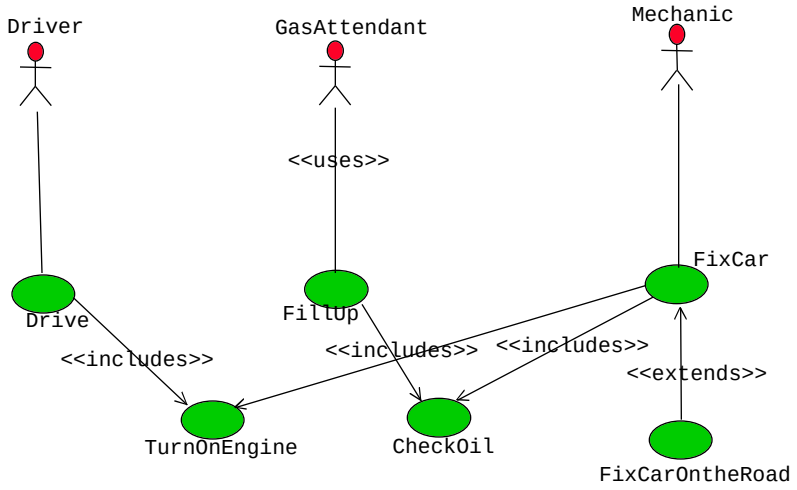
Documenting Use Cases

- For each use case, prepare a "flow of events" document, written from an actor's point of view.
- The document details what the system must provide to the actor when the use case is executed.
- Typical contents
  - ✓How the use case starts and ends;
  - ✓Normal flow of events;
  - ✓Alternate flow of events;
  - ✓Exceptional flow of events;

Use Cases for a Meeting Scheduling System



Use Cases for a Car



The State-of-Practice

- [Lubars93] reports on a field study involving 10 organizations; study's conclusions are as follows:
- Customer-specific projects are usually given large monolithic statements of requirements, seldom in machine-readable form; despite their size, these are often sketchy, ill-defined; concept of "super-designer" used for interpretation, filling gaps
- Market-specific projects often have smaller requirements, frequently produced in-house
- Securing customer interaction always hard
- Although many projects use document standards, few adopt any particular requirements method
- Projects using object-oriented methods had trouble modularizing their requirements; some thought that structured analysis led to unintelligible specifications; no uniform treatment of performance or real-time issues

The State-of-Practice

- General purpose tools (e.g., hypercard), general-purpose or special purpose CASE tools, were used with some success
- Many projects use some organizational approach to requirements validation
- About 1/3 of the projects did some sort of prototyping
- Organizational solutions preferred over technology -- instead of solving problems by buying hardware/software, organizations define procedures, assign people, set up committees etc.
- General-purpose technology preferred over CASE -- organizations rely more on word processors, hypertext, speadsheet products, rather than CASE tools
- Requirements activities are under-capitalized -- only 1/3 of the projects used some tools
- Market-driven projects increasingly important -- as the software market matures, organizations are buying a generic, off-the-shelve solution, rather than design a one-of system

Additional Readings

- [Easterbrook94] Easterbrook, S., "Resolving Requirements Conflicts", in [Jirotka94].
- [Booch99] Booch, G. et all. *The Unified Modeling Language User Guide*, Chapters 2, 16, 17. Addison-Wesley, 1999.
- [Jacobson92] Jacobson, I. et all. *Object-Oriented Software Engineering: A Use-Case Driven Approach*, Addison-Wesley, 1992.
- [Kotonya98] Kotonya, G. et all. *Requirements Engineering: Processes and Techniques*, John Wiley & Sons, 1998.
- [Lubars93] Lubars, M., Potts, C., Richter, C., "A Review of the State-of-Practice in Requirements Modelling", Proceedings, IEEE Symposium on Requirements Engineering, 1993.
- [Macaulay96] Macaulay, L., *Requirements Engineering*, Springer-Verlag, 1996..
- [Schneider98] Schneider, G. et all. *Applying Use Cases*, Addison-Wesley, 1998.